REDUX

STORE

Texto

Descripción generada automáticamente

Importamos las herramientas que necesitamos de las dependencias instaladas, como las devtools de redux que nos permiten tener más visibilidad de los procesos de redux que se van ejecutando a medida que realizamos algún tipo de accion, a su vez importamos un createStores, para crear el que será el almacenamiento de la “verdad absoluta” de los estados globales de redux y recibirá como parámetro todo los reducer, y el middleware a usar que será thunk, de redux-thunk. (explicacion)

INDEX DE CLIENT

Texto

Descripción generada automáticamente

envolvemos toda la aplicacio(componente App) con el provider que se encargara de entregarnos los valores asignados en los estados globales de redux, dentro del store

ACTIONS

Texto

Descripción generada automáticamente  
La función getAllCountries por medio de una función asíncrona obtiene todos los países de la base de datos enviados por el api y envía esos datos como payload y también envía un tipo de acción llamado “GET\_ALL\_COUNTRIES”. Y de la misma manera con la función getAllActivities.

Texto

Descripción generada automáticamente

Eso es posible solamente gracias a que se está usando el middleware Thunk. Nativamente redux no permite ningun tipo de acción asíncrona ni en el store, ni el reducer, ni en las actions. El thunk permite despachar una asincronía que con su response ejecuta a su vez una dispatch normal.

Dispatch

Las acciones tienen que ser enviadas al Store para que surgan efecto. La función dispatch base es un método de store, esta manda una acción de forma sincrónica a los reductores del store, junto con el estado previo retornado por el store, para calcular el nuevo estado. Espera que las acciones que les pasemos sean objetos planos, listas para ser consumidas por los reducers. También se puede envolver a los dispatchers en una serie de Middlewares, esto permite que los dispatchers puedan manejar acciones asincrónicas, además de poder transformar, demorar, ignorar o interpretar acciones antes de pasarla al siguiente Middleware.

Texto

Descripción generada automáticamente

En la función getCountryByName() recibe por parámetro el nombre del país a buscar y lo enviara por query, usamos un try catch para capturar errores, y enviara la data como payload, y enviara un type eso en caso exitoso, en caso de falla enviara como payload un arreglo con un id, name y una imagen de bandera que yo he seleccionado con mensaje de error 404 para informar al usuario.

En la función getById() recibirá por parámetro el id del país y lo enviara como params, la información obtenida de esa solicitud se enviara como payload, y se usará para mostrar la info de sus detalles.

Texto

Descripción generada automáticamente

La función resetDetailById() hace que al dar click en el home el valor del país en el estado actual sea vacio, porque sucedía que había como un bug al buscar los detalles de un país te mostraba el país anterior por un pequeño lapso de tiempo y luego cargaba el nuevo.

Función orderByName() y orderByPopulation() recibe parámetro el orden de ordenamiento y lo pasa como payload junto con un type

Function filterByContinent() y filterByActivity() recibe un parametro a evaluar y si cumple dicha condición se agregará en el estado de países a mostrar, dicho parámetro lo retornará como su payload y también con un type.

Texto

Descripción generada automáticamente

Función postActivity() recibe un parámetro en este caso llamado payload, usamos una función asíncrona, para ejecutar el método post en la url especificada dentro del primer parámetro del post y como segundo parámetro que será enviado por body ira el payload de los valores a crear en la base de datos.

Ya que el usuario será almacenado en la base de datos no será necesario realizar acciones en el reducer porque ese país será tomado cuando se despache la acción getAllCountries()

REDUCERS

Texto

Descripción generada automáticamente

Establecemos un estado inicial el cual se le pasara como parámetro igualándolo al estado de rootReducer, esto valores serán un estado de la data con los países la cual no se modificará y solo se obtendrá de la DB, se tendrá un valor para los países, las actividades, detalles, y países por continente.

…al hacer spring operator devolvemos el estado anterior. Un nuevo estado es generado, resultado de combinar el viejo estado y la acción del usuario. Este proceso lo realiza una función llamada reducer. Las describen que algo pasó, pero no especifican cómo cambió el estado de la aplicación en respuesta. Esto es trabajo de los reducers.

Texto

Descripción generada automáticamente

En el caso “ORDER\_BY\_NAME” usamos un operado ternario para validar el tipo de orden dispachado, para ordenar los elementos usamos el método sort donde se agregan dos parámetros comparadores que en este caso serán a y b, en caso de retornar un valor negativo quiere decir que el valor a va antes que el valor b y si devolvemos un positivo el valor b va antes que el valor a y si retorno el valor 0 los elementos se mantendrán en la misma posición.

Texto

Descripción generada automáticamente

Hacemos el ordenamiento con el método sort como previamente se había explicado y el resultado see convertirá en el valor del estado de countries.

Captura de pantalla de computadora

Descripción generada automáticamente

Filtrado por continente, si selecciono la opción de todos los continentes en países filtrados me pase la info de todos los países, en caso de elegir una opción distinta use el método filter() para filtrar todos los países que su continente coincida con con el enviado por el payload.

Texto

Descripción generada automáticamente

Si el payload es “All” el estado de countries serán todos los países. De lo contrario primero validadamos y filtramos los países con una actividad asociada. Luego en el forEach iteramos el array con países con actividad asociada donde recorremos todo el subarray de las actividades que contiene y validamos por medio del id si alguna de ellas coincide con la solicitada, que fue enviada por medio del payload, si es así pusheamos el país en un arreglo que al final ese arreglo será el valor del estado de countries.

Por default en caso de que el caso no haya sido similar a alguno de los especificados retornamos el estado actual.

COMPONENTES

Texto

Descripción generada automáticamente

BrowserRouter es un componente que nos entrega react-router-dom que nos permite establecer rutas en nuestra aplicación, según el componente que deseemos renderizar le especificamos en que ruta debe hacerlo, entonces dentro del componente principal de la aplicación lo envolvemos con BrowserRouter,

**BrowserRouter (Router)**: inyecta propiedades a nuestro componente para poder acceder al historial de navegación, realizar redirecciones, etc.

Usamos Switch que nos permite especificar que cuando encuentre cierta ruta renderice dicho componente, y para especificar la ruta usamos Route y le damos el un camino especificado usando path=””

Texto

Descripción generada automáticamente

Componente landing page, un texto de bienvenida, un enlace que te envía a la ruta home y una imagen de fondo. Usamos css modules porque a mi parecer me parece más cómodo de manejar y es muy similar a darle estilos y agregar la clase del estilo al css y html convencional además que me permite usar seudo clases como hover, focus y pseudo elementos como before y after.

Texto

Descripción generada automáticamente

En el componente Home importamos los hooks useState, UseEffect, useDispatch, useSelector, Link, los estilos de css a usar, también otros componentes a agregar y finalmente importamos las acciones a usar en el componente home.

Texto

Descripción generada automáticamente

Uso de useSelector() y useDispatch()

Para la paginación establecimos un estado de react para determinar en qué página estamos actualmente que inicialmente será 1, establecemos que queremos 10 elementos por página por defecto, lastIndex y firstIndex serán las variables encargadas de manejar que posiciones del arreglo se tomarán para ser mostradas en pantalla que irán de 10 en 10. Luego creamos una nueva variable llamada current countries donde irán los países según la página. Creamos un operador ternario donde diremos que si estamos en la página 1 muestre los primeros 9 elementos, de lo contrario mostrara 10 elementos, donde nos desplazaremos una posición atrás en los índices que habías establecido que serían múltiplos de 10.

Creamos una función llamada pagination que será pasada como parámetro al componente pagination, función la cual se encargara de setear el valor de la pagina en la que nos encontramos.

En la parte de abajo usamos un useEffect para decirle al componente que cada vez que se modifique el arreglo total de países me ubique en la página 1 de dichos elementos.

Texto

Descripción generada automáticamente

En el componente pagination recibimos por parámetro cantidad de países por página, el tamaño del arreglo con todos los países y una función pagination encargada de setear el estado que especifica en que página estamos.

El arreglo pageNumbers almacena el total de páginas, en un mismo arreglo. Luego validamos si dicho arreglo no está vacío vamos a separarlos en subArreglos más pequeños que se agruparan dentro de este con límite de máximo 10 páginas en cada sub-arreglo. Dentro de if hay un for que se encarga de agregar los 10 elementos, luego aumenta el índice en 10 y pushea los siguientes 10

Tenemos un useEffect que se encarga de estar atento si nosotros decidimos cambiar de posición en el arreglo esto por medio de los botones prev y next.

Texto

Descripción generada automáticamente

Tenemos las funciones handleNext y handlebefore, que nos mueven de una posición del arreglo de grupos a otra, luego especificamos que si hay elementos dentro el arreglo de grupos y la posición es diferente del valor 0 me renderice el botón de prev(ir atrás).

Renderiza las paginas y al hacer clic en algún numero este mostrara los países pertenecientes a esa página.

Luego usamos un ternario para preguntar si estamos en una pagina diferente a la ultima y si es el caso me renderice el botón de next(ir al siguiente grupo de páginas).

HOME

Texto

Descripción generada automáticamente

Usamos un useEffect component didMount para que al cargarse el componente me me traiga las actividades y los países, también un component diUpdate para que se actualicen los valores esos arreglos luego de cada dispatch, los dispatch se ejecutaran en las funciones especificadas cada vez que estas sean llamadas.

Texto

Descripción generada automáticamente

En la parte superior de la pagina tendremos estos elementos que nos permitirán organizar y filtrar los países según ciertas características de ellos. Tomamos dichas características y hacemos un dispatch pasado como parámetro el valor con el que deseamos filtrar u organizar.

Texto

Descripción generada automáticamente

En caso de que el arreglo de países este vacío muestre un componente de cargando, si no renderice los países traídos, en caso de que el país traído tenga un id de 404error me renderice una tarjeta con su data y un link que al darle clic recargue todo el sitio.

Finalmente, en la parte inferior tendrá l componente de paginado al cual le pasaremos por props cantidad de países por página, longitud total del arreglo de países y una función llamada pagination.

CARD

Texto

Descripción generada automáticamentecomponente que mostrara una tarjeta, con los valores del link de imagen de la bandera, nombre y continente. Valores que recibirá por props

SEARCH BAR

Texto

Descripción generada automáticamente

Este componente recibirá un valor en el input y al hacer clic en el botón de búsqueda despachará la función getCountrybyName().

CONTINENT FILTER

Texto

Descripción generada automáticamente

Tendremos un arreglo con todos los continentes.

Mapearemos esos valores del arreglo en un select y al seleccionar alguno de esas opciones en el select despachará la función filterByContinent() y como parámetro le pasará el continente seleccionado que su valor estará en su atributo html value=””. Tendremos un valor extra en el select llamado “all continents”, el cual traerá los países de todos los continentes de vuelta.

ACTIVITY FILTER

Texto

Descripción generada automáticamente

Por props recibimos un arreglo con todas las actividades registradas en la base de datos, mapeamos dicho arreglo y lo renderizamos como las options de un select, al hacer clic sobre alguna de las opciones este ejecuta una función que dispacha la acción filterByActivity() pasandole como parámetro el id de dicha acción.

DETAILS

Texto

Descripción generada automáticamente

importamos el hook useParams el cual nos permite leer la parte del enlace pasada como params, tomar ese valor y almacenarlo en una variable.

Usamos use selector para traer los detalles del país según su id, id que será tomado de params. Usaremos component didMount para que al montar el componente este ddespache la acción getById() pasandole como parámetro el id tomado de params. A su vez será un component didUpdate que estará atento a las modificaciones del id y dispatch por si se selecciona un país con un id diferente.

Tenemos una función de reset que al dar click en el Link de home para volver al componente Home despacha la acción resetDetailById() que se encargara de vaciar los valores del estado de detalles.

A su vez el valor del arreglo detalles contendrá las actividades vinculadas con el país las cuales se mapearán y se renderizarán en la parte derecha del de la tarjeta del país.

FORM ADD ACTIVITY

Texto

Descripción generada automáticamente

Creamos un estado de react en el cual almacenaremos los valores ingresados en cada campo del formulario, y creamos otro estado de react para almacenar los mensajes de error que mostraremos los cuales serán dos ya que solo dejaremos dos campos obligatorios. Y al final otro estado en el cual habilitaremos o deshabilitaremos el botón de enviar formulario en caso de que no haya ningún error.

Texto

Descripción generada automáticamente

Usaremos tres useEffect, el primero se encargará de estar atento a los cambios de los valores del estado de errores del formulario. Validamos que no exista ningún error en los valores del errorForm y si es el caso habilite el botón de enviar.

El segundo useEffect() se encarga de cargar todos los países.

Y el tercer useEffect valida si se ha elegido al menos un país de la lista, y en caso contrario de no hacerlo modifica el mensaje de error.

Texto

Descripción generada automáticamente

Creamos un arreglo con todas las opciones de duración, para luego mapearlas y mostrarlas en el select.

Creamos una función que valida que sean una modificación en el campo name, donde pondremos el nombre de la actividad y arroja un error o no según los valores ingresados, donde usaremos una expresión regular.

Texto

Descripción generada automáticamente

Tenemos dos funciones la cual handleChange se encaragar de setear los valores en el estado de formValues, esto según el atributo value del campo elegido. Y a su vez llamaremos la función validation y le pasaremos como parámetro la data del evento con lo cual ella validara si hay algún error o no en ello.

En la función handleSelectCount() tomamos el id del país seleccionado y lo agregamos a al arreglo.

Texto

Descripción generada automáticamente

En la función handleDelete() recibimos por parámetro el id del país sobre el que hicimos clic y hacemos un filtrado para ignorar ese elemento en el arreglo y de esta manera eliminarlo del arreglo countries\_id del estado de react de países a vincular con dicha actividad.

En la función handleSubmit() despachamos la acción postActivity() pasandole por parámetro los valores a enviar por body que se encuentra en estado llamado formValues y que se inserten en la base de datos según se estableció en dicha ruta, a su vez hacemos un reset de los valores en el estado para que luego de que se ejecute esa acción el formulario limpie los campos del formulario.

Hacemos un useSelector para traer el arreglo con todos los países en ese estado, almacenado en el store el cual no se modifica en absoluto.